L'Agilité du RAD

Comment adapter le RAD à l'Agilité

- Matthieu GIROUX - www.liberlog.fr
- Membre de www.caplibre.org
- Développeur Indépendant
- Installation et personnalisation Web
- Création de Logiciels de Gestion
- Création d'un savoir-faire

L'Agilité du RAD

Sommaire

1. Définitions et Histoire
2. Développement (Très) Rapide d'Applications
3. Programmation itérative de composants
4. Devenir éditeur avec du RAD ou VRAD

1.1) Définition : Rapid Applications Development

Rapid Applications Development ou RAD

=

Développement Rapide d'Applications ou DRA

= Créer visuellement pour créer vite

Le RAD permet de gagner du temps.

Le Développement Très Rapide d'Applications permet de créer votre logiciel de gestion d'entreprise personnalisé en sautant une étape dans la création du logiciel.

1.2) Mauvais exemples
Les API de développement

Une API est une bibliothèque à programmer

Les API de gestion nécessitent :

- De créer des sources difficiles à manipuler
- De programmer pour réinventer la roue

Un ingénieur développeur peut transformer des API de gestion afin de créer l'interface du logiciel de gestion à partir de certains fichiers de librairies VRAD comme LEONARDI.

1.3) Comment bien créer une librairie ?

Une seule règle : Eviter le copié-collé de code

Chronologie de création d'un savoir-faire :

- Au début on crée des unités de fonctions
- Puis on utilise et surcharge des composants
- On crée des paquets de composants
- On automatise les paquets en une librairie
- On ouvre alors sa librairie aux autres API
- La librairie nécessite peu de code ou aucun

1.4) Développement Rapide d'Applications (DRA ou RAD)

- Créer visuellement une application
- Pour gagner du temps dans la création
- Afin de créer une application intuitive

La plupart des outils RAD n'automatisent pas assez la gestion d'une entreprise.

Le Very Rapid Application Development est l'amélioration du RAD pour les serveurs de gestion ou d'autres interfaces définies.

2.1) Very Rapid Applications Development

Un logiciel est composé de :

- Une partie métier : Ce que veut le client
- Une partie technique : L'informatique

Que ce soit avec sans des outils RAD on :

- Mélangeait la technique et le métier
- Refaisait un logiciel entièrement

La partie métier du logiciel doit être gardée.

2.2) Le Développement Très Rapide d'Applications

Le VRAD permet de supprimer l'étape de programmation de l'interface homme-machine à partir de l'analyse.

Ce qui permet de créer l'interface homme-machine à partir de l'analyse avec des fichiers passifs.

Le VRAD est l'aboutissement du RAD et de la programmation par composants.

2.3) Fichiers passifs vs RAD classique

Avec les fichiers passifs on :

- Réfléchit fonctionnalités et coeur de métier
- Définit la présentation rapidement
- Détermine ce qui est faisable rapidement
- Détermine ce qui n'est pas modélisable
- Crée des plugins pour ce qui n'est pas fait
- Sait où l'on va

Le RAD ou VRAD permettent de travailler en amont du projet afin d'anticiper les futures étapes.

2.4) Intérêts du Développement Très Rapide d'Applications

Le Développement Très Rapide permet :

- D'empêcher mieux les erreurs de se produire
- De ne créer au final que l'analyse du logiciel
- De gagner du temps dans la création
- D'être indépendant de tout savoir-faire
- Que le programmeur pense fonctionnalités
- D'améliorer la qualité du logiciel créé

Le RAD et le VRAD sont intégrés dans les méthodes agiles.

2.5) Créer son interface avec des fichiers : le VRAD

Il est maintenant possible de créer une interface de gestion à partir de simples fichiers passifs.

Un fichier passif contenant la partie métier est lu et crée l'interface grâce au savoir-faire VRAD.

- GLADE GTK permet de créer une interface non liée aux données à partir de fichiers passifs.
- LEONARDI permet de créer une interface de gestion à partir de fichiers passifs.
- LIBERLOG possède un savoir-faire RAD en cours d'adaptation vers du VRAD.

2.6) Qualité VRAD

Avec un moteur VRAD :

- On gagne du temps et est plus agile
- On facilite la mise en place de futurs logiciels
- Ne teste que le moteur, pas l'interface créée
- L'analyse est le logiciel
- La maintenance est centralisée
- Le développeur va à l'essentiel

3.1) Programmation itérative de composants ?

La programmation orientée fonctionnalités permet

- De garder ce qui a été fait
- De centraliser un savoir-faire voire le partager
- D'utiliser réellement l'ingénierie en anticipant
- Que l'utilisateur puisse concevoir le logiciel avec l'analyste
- Que l'utilisateur soit mieux respecté
- De gagner du temps, de la fiabilité, de l'écoute

3.2) Programmation itérative de composants ?

Le composant permet de :

- Centraliser
- Adapter une micro-technique humaine
- Gagner du temps en RAD
- Faciliter le travail des développeurs en RAD
- Anticiper sur les demandes du client
- Eviter les copiés-collés
- Fiabiliser

3.3) Programmation itérative de composants – L'équipe

La programmation orientée fonctionnalités nécessite :

- Deux ingénieurs au moins pour les composants
- Des analystes pour un dialogue avec le client
- Des revues régulières avec le client
- Une direction souple, décisive et à l'écoute
- Un client décomplexé qui comprend comment est fait son logiciel pour savoir comment l'améliorer rapidement à sa guise

3.4) Programmation itérative de composants – La qualité

La programmation orientée fonctionnalités nécessite :

- Une conception simple, modulaire et générique
- Des tests unitaires sur les composants
- Du courage pour des composants évolutifs
- De la communication et des maquettes
- Un feedback pour anticiper sur les composants
- Des composants créés au moins à deux

3.5) Programmation itérative de composants – L'application

Les Programmeurs ou Analystes d'Applications :

- Demandent un dialogue permanent avec l'ingénierie et le client
- Doivent créer le logiciel facilement
- Font le minimum ou pas de programmation
- N'ont pas forcément besoin des tests unitaires

L'application est visible en maquette, inspectée, adaptée. La maquette deviendra le logiciel.

3.6) Programmation itérative de composants – Le composant

Un composant :

- Nécessite une structure cohérente
- Doit être lisible, organisé, simple, revu
- Possède le minimum d'objectifs
- Hérite de ce qui a été déjà fait
- Nécessite d'organiser en anticipant sur l'avenir
- Nécessite de la veille technologique
- Nécessite de disposer de toutes ses sources

3.7) Programmation itérative de composants – Les composants

Les composants :

- Sont créés avant la création du logiciel
- Sont regroupés autour de thématiques
- Sont modélisables facilement
- Représentent des micro-techniques humaines
- Permettent de créer des maquettes rapidement
- Permettent de gagner du temps en RAD

3.8) Programmation itérative de composants – Les programmeurs

Les sources du logiciel ou des composants :

- Appartiennent à tous les programmeurs
- Possèdent au début de chaque page les auteurs ayant participé
- Nécessitent les noms pour les modifications
- Nécessitent des revues pour les bons points et les mauvais points, les avancées et les reculs
- Nécessitent de savoir où l'on va

4.1) Développement Rapide vs Ligne de commande

Exemple : Création d'une fiche HTML simple

Un code centralisé utilisé avec du copié-collé

- 3 jours et ça n'est peut-être pas fini

La même chose avec un outil RAD

- ½ journée d'analyse et ½ journée de création
- Le composant automatise certaines créations
- La fiche est utilisable sans avoir trop à tester

4.2) Fichiers passifs VRAD vs RAD classique

Les fichiers passifs :

- Permettent de définir le coeur de métier en eux
- Peuvent être créés à partir d'une analyse
- Rendent indépendants du savoir-faire utilisé
- Sont définis et peuvent évoluer
- Permettent de créer d'autres interfaces
- Permettent de penser fonctionnalités

C'est l'analyse d'½ journée qui crée le logiciel.

L'analyse correspond au logiciel créé.

4.3) Devenir éditeur

Il est temps de se poser des questions. Afin de devenir éditeur et de fidéliser ses clients :

- Diffuser son savoir-faire à beaucoup de clients permet l'autonomie
- Automatiser son savoir-faire permet de satisfaire ses clients ou futurs clients
- Utiliser l'agilité renforce la confiance du client
- Le RAD et le VRAD autorisent une élite avec des analystes orientés communication